Opinnäytetyö (AMK)

Tietojenkäsittelyn koulutusohjelma

Tietojärjestelmät

2014

Mitra Tanhai

DYNAAMISEN KÄYTTÖLIITTYMÄN SUUNNITTELU JQUERYLLA

TURUN AMMATTIKORKEAKOULU
TURKU UNIVERSITY OF APPLIED SCIENCES

OPINNÄYTETYÖ (AMK) | TIIVISTELMÄ

TURUN AMMATTIKORKEAKOULU

Tietojenkäsittelyn koulutusohjelma | Tietojärjestelmät

Joulukuu 2014 | 29 sivua

Anne Jumppanen

Mitra Tanhai

DYNAAMISEN KÄYTTÖLIITTYMÄN SUUNNITTELU JQUERYLLA

Tässä opinnäytetyössä käsitellään verkkosovellusten dynaamisia käyttöliittymiä ja niiden suunnittelua. Työn toimeksiantona oli NEMO-projektin sosiaalisen median louhintatyökalun ja big data –tietokannan dynaamisen käyttöliittymän suunnittelu ja toteutus. Työn tavoitteena oli luoda toimiva ja helppokäyttöinen käyttöliittymä jQueryn avulla.

Dynaaminen suunnittelu on tuonut mahdollisuuden vuorovaikutteisempien verkkosivustojen toteuttamiseen. Verkkosivuista halutaan yhä enemmän tietokoneohjelmien kaltaisia ja käyttäjän kanssa paremmin keskustelevia. Dynaamisuuden avulla verkkosovelluksille on saatu enemmän suorituskykyä ja parempaa käytettävyyttä.

JQuery on ilmainen, nopea ja monipuolinen JavaScript-kirjasto, jonka avulla voidaan luoda monimutkaisiakin toteutuksia vain muutamalla rivillä koodia. Se on tehokas apu dynaamiseen web-suunnitteluun ja tarjoaa ratkaisua selainten yhteensopivuusongelmiin. Koodikirjasto kattaa HTML/DOM ja CSS –manipuloinnin, tehosteet, tapahtumat, animaatiot ja AJAXin.

ASIASANAT:

Dynaamisuus, käyttöliittymäsuunnittelu, jQuery, käytettävyys

BACHELOR´S THESIS | ABSTRACT

TURKU UNIVERSITY OF APPLIED SCIENCES

Information Technology | Information Systems

December 2014 | 29 pages

Instructor Anne Jumppanen

Mitra Tanhai

DYNAMIC INTERFACE DESIGN WITH JQUERY

The purpose of this thesis was to examine how to design dynamic interfaces for web applications. The goal of the practical part was to design and create a dynamic web interface for the social media miner and big data database of the NEMO project. The objective was to create a functional and user-friendly interface with jQuery.

Dynamic web design has brought an opportunity to create more interactive web pages. The current users require more interaction and characteristics of a desktop application software from web applications. Using dynamic design also improves the performance and usability of the web application.

JQuery is a free, fast and versatile JavaScript library, which helps create complex implementation with only few lines of code. It is a powerful tool for dynamic web design and helps solve browser incompatibility problems. The library consists of HTML/DOM and CSS manipulation, effects, events, animations and AJAX.

KEYWORDS:

Dynamic, interface design, jQuery, usability

SISÄLTÖ

KUVAT

KÄYTETYT LYHENTEET

AJAX	Asynchronous JavaScript And XML
CSS	Cascading Style Sheets
DOM	Document Object Model
HTML	Hypertext Markup Language
LAMP	Linux, Apache, MySQL, PHP
SOME	Sosiaalinen media
SQL	Structured Query Language
XML	Extensible Markup Language

1 JOHDANTO

Opinnäytetyössä tutustutaan dynaamisiin verkkosivuihin, niiden toteuttamiseen jQuerylla sekä käyttöliittymän suunnitteluun. Toimeksiantona tähän työhön toimi NEMO-projektin sosiaalisen median louhintatyökalun ja big data –tietokannan dynaaminen käyttöliittymä. Käyttöliittymän avulla voidaan hakea tietoja big data -tietokannasta sääntöjä asettamalla.

NEMO on yhteistyöhanke Turun ammattikorkeakoulun, Turun yliopiston kauppakorkeakoulun ja Tampereen teknillisen yliopiston välillä. Mukana on myös parikymmentä erikokoista yritystä sekä kuluttaja- ja yritysmarkkinoilta että palvelu- ja tuotantoaloilta. Projekti on tarkoitus suorittaa vuosien 2014-2015 aikana.

Dynaamiset sivut ovat tuoneet mahdollisuuden vuorovaikutteisempien verkkosivujen toteuttamiseen. Verkkopalveluista halutaan yhä enemmän tietokoneohjelmien kaltaisia ja käyttäjän kanssa paremmin keskustelevia. Dynaamisuus on myös parantanut sivustojen tietoturvaa, sillä toiminnallinen lähdekoodi ei ole enää nähtävissä pelkällä selaimella.

2 KÄYTTÖLIITTYMÄN SUUNNITTELU

Käyttöliittymällä tarkoitetaan rajapintaa, jolla käyttäjä käyttää laitetta, ohjelmistoa tai tuotetta. Melkein kaikilla elektronisilla laitteilla on käyttöliittymä, ja se voi olla hyvinkin yksinkertainen, kuten esimerkiksi digitaalisessa kellossa, tai monimutkainen kuten tietokoneohjelmassa. Käyttöliittymää voi olla joskus vaikea erottaa laitteesta, siksi ne on suunniteltava ja arvioitava yhtenä kokonaisuutena.

Tietokoneissa käytetään nykyään lähes aina graafista käyttöliittymää merkkipohjaisen sijaan. Kosketusnäyttöiset laitteet ovat lähiaikoina saavuttaneet paljon suosiota, ja se on otettava huomioon käyttöliittymiä suunnitellessa. Erilaisten mobiililaitteiden yleistyessä myös näyttökoko vaihtelee suuresti pienestä matkapuhelimen näytöstä pöytätietokoneen laajakuvanäyttöön.

Käyttöliittymän tulee palvella kohderyhmäänsä mahdollisimman hyvin ja tehokkaasti, ja suunnittelun tuleekin aina lähteä kohderyhmän vaatimuksista. Esimerkiksi tietokoneella aloittelijoiden ja aktiivikäyttäjien välillä on suuri ero. Kohderyhmä vaikuttaa kaikkeen käyttöliittymän ominaisuuksia suunnitellessa.

Käyttöliittymän rakenne perustuu yleensä neljään peruselementtiin: ikkunat, valikot, ikonit ja ohjauslaitteet. Ikkuna on alue, jossa näytetään käyttäjän valitsemat tiedot, kuten dokumentti. Ohjelmat avautuvat yleensä omiin ikkunoihinsa, joissa voidaan navigoida ohjelman omien valikoiden kautta. Valikot sisältävät yleisimpiä käskyjä, joita sovelluksessa voidaan käyttää. Ikonit eli kuvakkeet ovat pieniä kuvia, joiden on tarkoitus ilmaista käyttäjälle, mikä on niiden tarkoitus ja toiminta. Ne ovat yleensä käyttäjille tuttuja symboleja tai logoja, esimerkiksi verkkosivulla yrityksen Facebook-profiiliin vievä painike tai videon toistosivulla play- ja pause-painikkeet. Ohjauslaitteilla tarkoitetaan tapoja, joilla ohjataan sovelluksen toimintaa, esimerkiksi hiiri, näppäimistö tai kosketusnäyttö. (Salovaara 2011, 6).

Ohjelmistojen ja internet-sivujen käyttöliittymäsuunnittelu eroavat hieman toisistaan, mutta niitä kumpaakin tulisi yhdistää helppokäyttöisyys, yhtenevyys, ja toimivuus. Tässä työssä keskitytään käyttäjän ja verkkopalvelun väliseen käyttöliittymään ja sen suunnitteluun.

2.1 Käytettävyys

Hyvän käyttöliittymän laatukriteerinä toimii käytettävyys. Käytettävyys on määritelty ISO 9241-11 –standardissa seuraavasti: "Se vaikuttavuus, tehokkuus ja tyytyväisyys, jolla tietyt määritellyt käyttäjät saavuttavat määritellyt tavoitteet tietyssä ympäristössä". Käytettävyys voidaan standardin mukaan määritellä myös riippuvaksi käyttötilanteesta.

Käytettävyys on menetelmä- ja teoriakenttä, jonka avulla pyritään parantamaan käyttäjän ja käyttöliittymän yhteistä toimintaa ja saamaan se mahdollisimman miellyttäväksi käyttäjälle. Kognitiivista psykologiaa sekä ihmisen ja koneen vuorovaikutusta voidaan käyttää avuksi käytettävyyden parantamisessa. Tuotteen käytettävyys on kuitenkin vain yksi osa sen käyttökelpoisuudesta. (Sinkkonen ym. 2006, 17).

Hyvän käytettävyyden tunnusmerkkejä ovat opittavuus, käytön tehokkuus, muistettavuus, virheiden vähyys ja subjektiivinen miellyttävyys. Kun tuote tai palvelu on hyvin suunniteltu, siihen perehtymätönkin ymmärtää miten sitä käytetään. Tuotteen tulisi tukea mahdollisimman hyvin niitä tehtäviä, joihin se on tarkoitettu, ja tehtävien tulisi sujua paremmin tai miellyttävämmin kuin ilman tuotetta. (Sinkkonen ym. 2006, 25).

Ihmiset ovat tottumuksiltaan ja tuotteiden käyttäjinä hyvin erilaisia, mutta jotkin ihmisen ominaisuudet eivät juurikaan muutu. Näiden ominaisuuksien tunteminen voi auttaa parempien tuotteiden suunnittelussa. Suhteellisen pysyviä asioita ovat toimintakulttuurin opitut asiat, kulttuurin omaksutut ominaisuudet ja synnynnäiset ominaisuudet. Muuttuvia asioita taas ovat persoona- ja tilannekohtaiset asiat sekä vaihtelevat kulttuurielementit, esimerkiksi muoti, alakulttuurit ja työpaikkakohtaiset toimintatavat. (Sinkkonen ym. 2006, 24-25).

Käyttöliittymän vuorovaikutuksen rakentamisessa on visuaalisella suunnittelulla suuri merkitys. Ulkoasun tulee tukea tuotteen käsitteellistä sisältöä ja luoda tuotteesta yhtenäinen kokonaisuus. Selkeä ulkoasu vastaa käyttäjän kokemusta todellisuudesta ja auttaa tärkeiden signaalien näkyvyydessä. Tällöin myös aloittelevat käyttäjät hahmottavat helpommin kokonaisuuden.

Web-sivujen kieli koostuu pääasiassa tekstistä, kuvista ja symboleista. Käyttäjät ovat oppineet tietyntyylisten symbolien tarkoittavan tietynlaista toimintamahdollisuutta, ja toimivat sen mukaan katsoessaan käyttöliittymää. Symbolikielen puutteita voidaan korvata web-suunnittelussa selitysteksteillä ja ohjeilla. Täsmällisempiä symbolirakenteita tarvitaan esimerkiksi elektronisten laitteiden pienillä näytöillä toimiviin käyttöliittymiin, joiden vuorovaikutus toimii pitkälti valikoiden ja näppäinten varassa. (Sinkkonen ym. 2006, 109).

Käytettävyyden kannattaa perustua intuitioon ja käyttäjän jo oppimiin asioihin. Käyttäjä on tottunut joihinkin asioihin käyttöliittymässä, esimerkiksi siihen että verkkosivulla logoa napsauttamalla pääsee takaisin etusivulle. Näitä opittuja asioita kannattaa siis hyödyntää, eikä lähteä liikaa muuttamaan käyttäjää hämmentävillä tavoilla. (Sonninen 2011, 12).

Käyttöliittymän käyttölogiikkaa voidaan rakentaa myös asettamalla rajoituksia. Rajoituksilla voidaan vähentää käyttäjän toimintamahdollisuuksia ja estämällä siten väärän vaihtoehdon käyttäminen. Rajoitukset voivat olla fyysisiä, loogisia tai kulttuurisia. Fyysinen rajoitus voi esimerkiksi olla painike, jota voidaan painaa ainoastaan kun se on sallittu. Kun käyttäjä itse päättelee, mikä on käytössä, on kyseessä looginen rajoitus. Kulttuurinen rajoitus on jokin asia jonka olemme kulttuurissamme tottuneet tekemään tietyllä tavalla. (Sinkkonen ym. 2006, 136).

Kulttuurisille rajoituksille sukua oleva konventio on tärkeä asia suunnittelussa. Konventio tarkoittaa tuttua asiaa jonka olemme oppineet tekemään samalla tavalla samantyyppisissä laitteissa. Hyvät käytännöt siirtyvät nopeasti konventioiksi, joita ihmisten on helppo oppia käyttämään. Konventioita kannattaa hyödyntää käyttöliittymien suunnittelussa, mutta niitä voi myös luoda lisää. Hyvä tapa on yhdistää vanhoja ja uusia asioita. Konventioiden kohdalla kannattaa tosin aina selvittää jakavatko käyttäjät samat konventiot kuin suunnittelija itse. (Sinkkonen ym. 2006, 136).

2.2 Responsiivinen suunnittelu

Käyttöliittymien suunnittelu ei ole enää yhtä helppoa kuin ennen, sillä verkkosovelluksia ja sivustoja käytetään yhä useammin jollain muulla, kuin perinteisellä pöytätietokoneella. Erilaiset tablettitietokoneet ja älypuhelimet ovat yleistyneet, ja näytön koko vaihtelee suuresti. Suurelle näytölle suunniteltu sivusto ei toimi kovin käytännöllisesti pienellä kosketusnäytöllä, ja eri sivustot eri laitteille ovat raskaita ja aikaa vieviä ylläpitää. Tässä avuksi tulee responsiivinen suunnittelu.

Responsiivinen suunnittelu tarkoittaa sitä, että verkkosivu skaalautuu eli mukautuu sopivaan kokoon eri päätelaitteilla katsottaessa. Responsiivisuus huomioi päätelaitteen näyttökoon asettamat rajoitukset ja käyttäjä saa aina parhaan mahdollisen käyttökokemuksen päätelaitteesta riippumatta. Sisällön sommittelu toteutetaan yleensä määrittelemällä elementtien mitat prosentteina suhteessa käytettävissä olevaan tilaan.

Mukautuvan käyttöliittymän suunnittelu on haastavampaa kuin perinteisen staattisen käyttöliittymän, mutta yleensä se kannattaa. Se parantaa käyttökokemusta sekä navigoinnin että näkymän kannalta ja sivusto palvelee eri käyttäjäryhmiä paremmin. Sivuston brändi-ilme pysyy johdonmukaisena päätelaitteesta riippumatta.

Sivujen asemointi suunnitellaan useassa eri muodossa, joten prosessi vaatii suhteessa enemmän aikaa sivuston arkkitehtuurin suunnitteluun ja testaamiseen. Kaikissa laitteissa on näytettävä oleellinen sisältö huomioiden näkymän

koko, käyttötilanne ja interaktiomenetelmä. Suunnittelu voidaan jakaa karkeasti seuraaviin vaiheisiin:

- mukautuvan verkkosivuston lisävaatimusten selvittäminen ja sivujen sisällön kartoittaminen
- sisällön kuten kuvien ja tekstin määrittely ja priorisointi eri laitteille
- layoutin suunnittelu rautalankamalleina eri resoluutioille
- tyylitiedostojen huomioiminen
- prototyyppi (HTML-kieli, CSS-tiedosto tai Media Query -säännöstö) (Karukka & Inkilä 2014).

Eri koodien eri laitteille kirjoittamisen sijaan responsiivisessa suunnittelussa käytetään laitteiden ja selainten ominaisuuksien tunnistamista ja niihin mukautumista. Kyse ei ole vain fyysisistä laitteista vaan niiden erilaisista käyttötavoista, esimerkiksi tablettitietokoneen pitämisestä pysty- tai vaakasuunnassa tai pöytäkoneen selainikkunoiden koon muuttamisesta. Tärkeintä on selaimen käytössä oleva koko. (Korpela 2012).

Responsiivisuus voidaan toteuttaa JavaScriptillä, mutta nykyaikaisempi ja suositeltavampi tapa on käyttää CSS:n mediarajoituksia (media queries). Mediarajoituksilla voidaan asettaa eri muotoilut erikokoisille näytöille, yleensä asettamalla ikkunan minimi- ja maksimileveys. Esimerkiksi kuvassa 1 asetetaan mediarajoituksella sivun taustaväri vaaleansiniseksi ikkunassa, jonka maksimileveys on 300px. Elementtien sijoittelu tapahtuu asemoinnin (positioning) tai kellutuksen (floating) avulla. Ikkunoita ei yleensä määritellä vain yhteen kokoon, vaan tietyt rajoitukset täyttävään kokoon. Mediaehdoilla voidaan myös selvittää, onko näyttö pysty- vai vaakasuunnassa. (Korpela 2012).

```
@media screen and (max-width: 300px) {
    body {
        background-color: lightblue;
    }
}
```

Kuva 1. CSS:n mediarajoitus jossa ikkunan maksimileveys on 300px.

3 DYNAAMISET VERKKOSIVUT

Verkkosivu voi olla joko staattinen tai dynaaminen. Staattinen sivu on palvelinkoneella oleva muuttumaton tiedosto, joka noudetaan selaimeen. Se muuttuu ainoastaan, jos palvelimella olevaa tiedostoa muokataan. Dynaaminen verkkosivu taas muodostetaan vasta silloin, kun selain kutsuu sitä. (How does the Internet work 2014).

Dynaaminen verkkosivu saattaa olla jokaisella vierailukerralla erilainen. Tämä mahdollistaa muun muassa erilaiset käyttäjän syöttämät parametrit, jotka voivat vaikuttaa sivuun tai hakuajankohdasta riippuvat toiminnallisuudet, esimerkiksi päivämäärän ja kellonajan näyttämisen. Sivuston turvallisuutta parantaa se, että tiedon muodostumismekanismi voidaan piilottaa käyttäjältä. Selaimen avulla lähdekoodin katsominen ei paljasta, miten dynaaminen tieto muodostuu. (Ajax Introduction 2014).

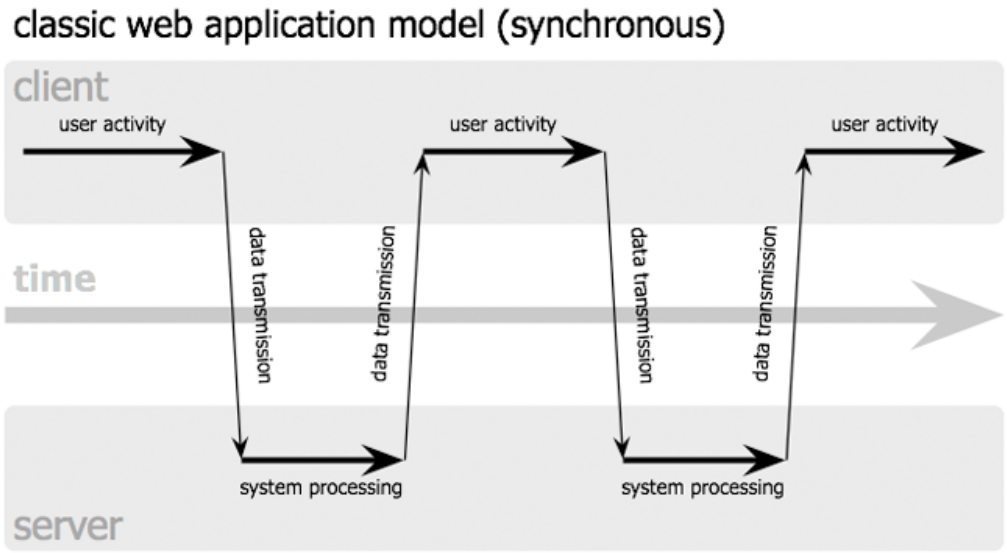
3.1 Ajax

Ajax eli Asynchronous JavaScript and XML tarkoittaa joukkoa web-sovelluskehityksen tekniikoita, joilla voidaan rakentaa dynaamisuutta verkkosivulle. Ajaxin avulla selain vaihtaa tietoa palvelimen kanssa ilman, että koko sivua tarvitsee ladata uudelleen, kun käyttäjä tekee muutoksen. Tämä parantaa vuorovaikutusta käyttäjän ja ohjelman välillä sekä lisää nopeutta ja käytettävyyttä.

Ajax muodostuu XHTML/HTML, CSS, DOM, XML ja XMLHttpRequest -tekniikoiden yhdistelmästä. XHTML/HTML ovat web-merkintäkieliä ja CSS:ää käytetään muotoiluun. DOM eli Document Object Model tarkoittaa ohjelmointirajapintaa, jota tässä tapauksessa käytetään informaation vuorovaikutukseen ja dynaamiseen esittämiseen. XMLHttpRequest-objektin tehtävä on vaihtaa tietoa palvelimen kanssa asynkronisesti. XML:ää käytetään formaattina tiedonvälitykseen palvelimen ja verkkosivun välillä. (Ajax Introduction 2014).

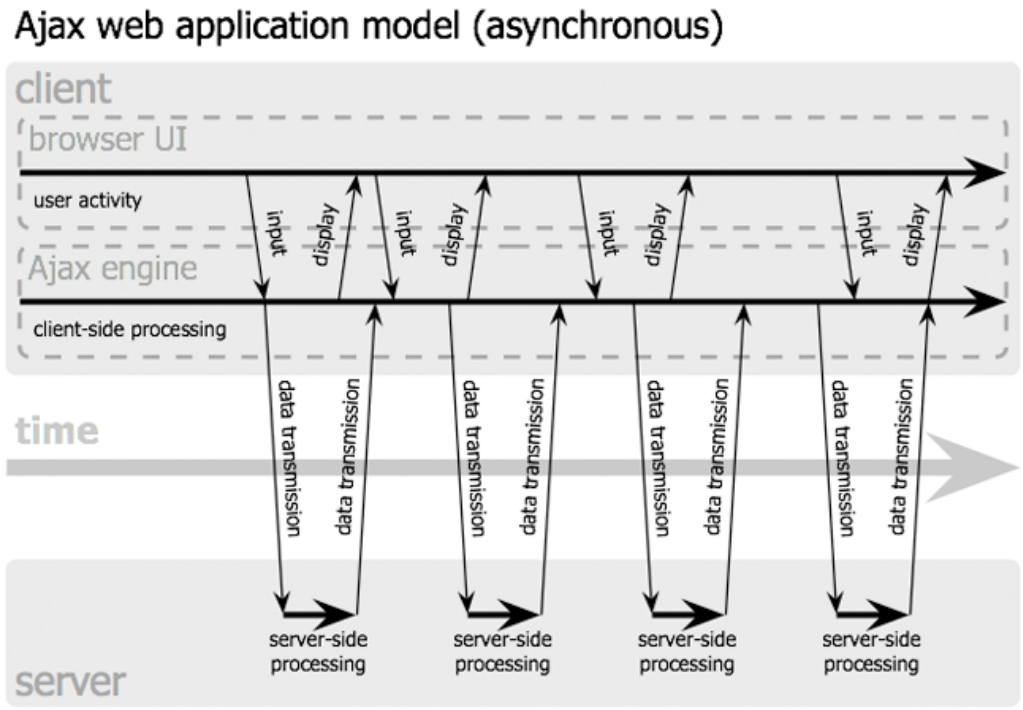
Ajaxin avulla verkkosivuista saadaan dynaamisempia ja kevyempiä. Ajaxia hyödyntämällä verkkosivuista on saatu enemmän työpöytäsovelluksen kaltaisia, niillä on samankaltainen ulkoasu, suorituskyky ja käytettävyys. Suorituskyky paranee, kun Ajax pystyy erottamaan käyttäjän tekemät toiminnot ja palvelimen kommunikaation suorittaen niitä rinnakkain. Näitä sovelluksia kutsutaan termillä RIA eli Rich Internet Application. (Rainio 2011, 6).

Jos tarkastellaan perinteisten staattisten verkkosovellusten toimintaa, käyttäjän toiminta saa aikaan HTTP-pyynnön palvelimelle, joka vastaa palauttamalla tiedoston (kuva 2). Tämä on muuten toimiva malli, mutta se vie aikaa ja koko sivu ladataan joka kerta uudelleen vaikka muutokset olisivat hyvin pieniä. Tämä rasittaa siirtokaistaa ja palvelinta. (Garret 2005).



Kuva 2. Synkronisen verkkosovelluksen vuorovaikutus. (Garret 2005).

Ajax luo erillisen kerroksen käsittelemään kommunikointia palvelimen ja käyttöliittymän välillä (kuva 3). Tämä toiminta on asynkronista, eli käyttäjän ei tarvitse jäädä odottamaan vastausta palvelimelta. Käyttäjä voi tehdä jatkaa toimintaansa häiriintymättä ja lähettää lisää pyyntöjä, jotka käsitellään järjestyksessä taustalla. Paluudata latautuu käyttöliittymään ilman koko sivun uudelleenlatausta.



Kuva 3. Asynkronisen verkkosovelluksen vuorovaikutus. (Garret 2005).

3.2 JQuery

jQuery on JavaScript-kirjasto, joka helpottaa JavaScriptin käyttöä ja websovellusten luomista. Se on nopea, monipuolinen ja tehokas apu websuunnitteluun ja tarjoaa ratkaisua ongelmiin, jotka liittyvät muun muassa selainten yhteensopivuuteen ja DOM-solmujen käsittelyyn. (JQuery Introduction 2014).

JQueryn käyttö on ilmaista. Jo valmiiksi laajaan kirjastoon on saatavilla runsaasti kolmannen osapuolen liitännäisiä ja omien funktioiden kirjoittaminen on myös mahdollista. JQuery helpottaa koodin kirjoittamista, sillä toiminnallisuus joka tavallisesti vaatisi monta riviä koodia voidaan toteuttaa muutamalla rivillä. JQuery on laajuutensa ja mukautettavuutensa vuoksi hyvä työkalu dynaamisten verkkosivujen toteutukseen.

Vaikka jQuery on suhteellisen uusi tulokas JavaScript-kirjastojen maailmassa, se on ansainnut suurten yritysten tuen. JQuerya käyttävät muun muassa yritykset kuten IBM, Netflix, Amazon, Dell ja Twitter. Microsoft on myös ilmoittanut käyttävänsä jQuerya Visual Studio –työkaluissaan ja Nokia matkapuhelimissaan. (Bibeault & Katz 2010, 4).

Koodikirjasto kattaa HTML/DOM ja CSS –manipuloinnin, tehosteet, tapahtumat, animaatiot ja AJAXin. JQueryn käyttöönotto tapahtuu lataamalla kirjasto jQueryn verkkosivuilta tai käyttämällä CDN (Content Delivery Network) -palvelua, esimerkiksi Googlea. JQuery-kirjasto on yksi JavaScript-tiedosto, jota voidaan kutsua HTML-sivun <head>-tagissa.

Kuten CSS pyrkii erottamaan tyylin HTML-sivun rakenteesta, jQuery pyrkii erottamaan toiminnan rakenteesta. Strategiaa kutsutaan huomaamattomaksi JavaScriptiksi (Unobtrusive JavaScript), ja sen toivat esille jQueryn tekijät. Sen jälkeen kyseistä tekniikkaa on käytetty lähes kaikissa suurimmissa JavaScript-kirjastoissa. Erottelu parantaa verkkosivuston koodin organisointia ja monipuolisuutta. (Bibeault & Katz 2010, 6).

JQueryn syntaksi on muotoa ***$(selector).action()***, jossa $-merkki merkitsee koodin jQueryksi, ***selector*** valitsee HTML-elementin ja ***action*** tapahtuman joka suoritetaan valitulle elementille. Esimerkiksi koodi ***$("p").hide()*** piilottaa kaikki <p>-elementit sivulta. (JQuery Introduction 2014).

Vaikka jQuery tarjoaakin hyvät työkalut vastaamaan suurinta osaa websuunnittelun tarpeista, voidaan joskus tarvita muita JavaScript-kirjastoja jQueryn rinnalla. Tällainen tilanne voi tulla esimerkiksi toista kirjastoa käyttävän sovelluksen siirtämisessä jQueryn pariin. JQuery on rakennettu niin, ettei se suurilta

osin ole konfliktissa muiden kirjastojen merkintätavan kanssa, mutta joskus esimerkiksi $-merkki voi tuottaa ongelmia. JQueryn tekijät eivät ole halunneet sulkea pois muiden kirjastojen käyttämistä, joten konfliktitilanteita varten on kehitetty noConflict()-funktio, joka korjaa ongelman. (Bibeault & Katz 2010, 16).

3.3 JQuery UI

JQuery UI on virallinen laajennus jQuery-kirjastoon, ja se sisältää kokoelman komponentteja graafisen käyttöliittymän toteuttamisen avuksi. Se koostuu kolmenlaisista elementeistä:

- **Interaktiot** (*interactions*) – kursorilla tapahtuva vuorovaikutus, kuten elementin siirtely, koon muuttaminen, valitseminen ja lajittelu.
- **Komponentit** (*widgets*) – yleisimpiä lisäpalikoita käyttöliittymään, esimerkiksi päivämäärävalitsin, edistymispalkki tai valikko.
- **Tehosteet** (*effects*) – edistyneempiä tehosteita kuin ydinkirjastossa, muun muassa animaatioita ja luokkamuunnoksia.
  (Bibeault & Katz 2010, 282).

JQuery UI on varsin laaja kirjasto, eikä käyttäjä välttämättä tarvitse kaikkia tarjolla olevia elementtejä. Tämän vuoksi lataussivulla on tarjolla mahdollisuus rakentaa kirjasto juuri niistä elementeistä kuin käyttäjä haluaa. On turhaa ladata suurempi määrä koodia kuin sovelluksessa aiotaan käyttää. Latauksen mukana tulee kansiot CSS- ja JavaScript-tiedostoille sekä kokoelma dokumentteja, kuten demoja, lisenssejä ja dokumentaatioita. (Bibeault & Katz 2010, 282).

HTML:n tarjoamat kontrollit ovat hyvin suppeat eikä parempien ohjelmointi ole yksinkertaista. JQuery UI tarjoaa helpotusta yleisiin ongelmiin peruskontrollien kanssa, kuten syötteiden hallintaan lomakkeessa, sisällön järjestämiseen tai päivämäärän valitsemiseen kalenterista. JQuery UI sisältää useita valmiita komponentteja, jotka ovat käteviä kun elementtien pitäisi sekä näyttää hyvältä että toimia luotettavasti. (Bibeault & Katz 2010, 347).

Kun tarvittavat tiedostot on ladattu, komponenttien käyttäminen verkkosivulla on helppoa. Käyttäjän tulee aluksi lisätä HTML-sivun koodiin viittaukset skripti- ja CSS-tiedostoihin, jonka jälkeen mitä tahansa jQuery UI:n komponenttia voidaan käyttää sivuilla (kuva 4).

```
1 | <link rel="stylesheet" href="jquery-ui.min.css">
2 | <script src="external/jquery/jquery.js"></script>
3 | <script src="jquery-ui.min.js"></script>
```

Kuva 4. HTML-dokumenttiin lisättävät viittaukset.

Jos käyttäjä haluaa ottaa käyttöön esimerkiksi tässäkin opinnäytetyössä käytetyn päivämäärävalitsimen, tulee hänen vain lisätä datepicker-funktion kutsu sivun HTML- ja JavaScript tiedostoihin (kuva 5). Komponenttien asetukset ovat hyvin muokattavissa, ja datepicker-funktiolle voidaan asettaa muun muassa oletuspäivämäärä, formaatti ja rajoituksia saatavilla oleviin päivämääriin. Lopputuloksena saadaan kenttä, johon hiirellä napsauttaessa avautuu kalenteri päivämäärän valitsemista varten (kuva 6).

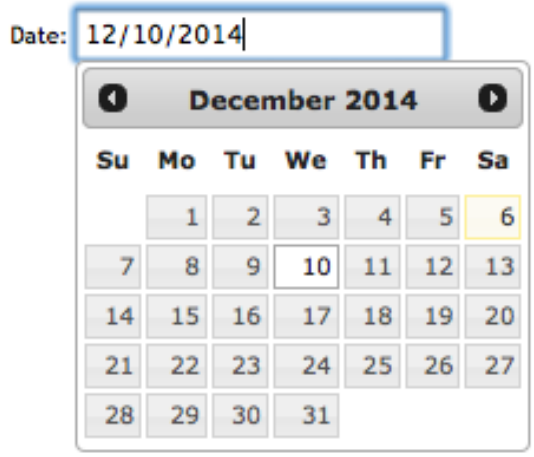
HTML:

```
1 | <input type="text" name="date" id="date">
```

JavaScript:

```
1 | $( "#date" ).datepicker();
```

Kuva 5. Datepicker-funktion koodit HTML- ja JavaScript-tiedostoissa.



Kuva 6. Valmis päivämäärävalitsin.

4 SOME-LOUHIJAN JA BIG DATA –TIETOKANNAN DYNAAMINEN KÄYTTÖLIITTYMÄ

Tämän opinnäytetyön toimeksiantona on NEMO-projektin sosiaalisen median big data -tietokannan ja Twitter/Facebook-louhintaohjelmiston rajapinnan dynaamisen web-käyttöliittymän toteutus.

Sosiaalisen median louhintaohjelmiston tehtävänä on hakea viestejä muun muassa Facebookista ja Twitteristä myöhempää analysointia varten. Dynaamisen käyttöliittymän on tarkoitus olla mahdollisimman käyttäjäystävällinen, ja sen avulla voidaan louhia tietoa big data -tietokannasta sääntöjä asettamalla. Säännöt rajaavat tai/ja analysoivat hakua ja tuottavat datasetin lopputulokseksi. Käyttäjä voi esimerkiksi hakea johonkin henkilöön liittyvät Facebook-viestit viimeisimmän viikon ajalta yhdistelemällä hakusanoja AND-, OR- ja NOT-operaattoreilla.

NEMO-projekti tutkii miten ristiriitaisia ja negatiivisia tunteita voidaan hyödyntää eettisesti kestävällä tavalla asiakaskokemusten ja työilmapiirin parantamisessa, yhteiskuntavastuun viestimisessä sekä kasvun, innovaatioiden ja uusien liiketoimintamallien lähteenä. Turun AMK on koordinaatiovastuussa osahankkeeseen nimeltä ”Wake Up and Smell the Coffee!” – Tunnetaidot sosiaalisessa mediassa. Sen tavoitteena on selvittää, miten sosiaalisessa mediassa käytävää negatiivisia ja ristiriitaisia tunteita herättävää keskustelua voidaan hyödyntää asiakaskokemuksen parantamisessa sekä kehitettäessä tuotteita ja palveluita.

4.1 Suunnitteluprosessi

Suunnittelu aloitettiin keskustelemalla NEMO-projektin jäsenten kanssa tulevan käyttöliittymän tarpeista ja luomalla vaatimusmäärittely. Lähtötilanne oli se, että nykyinen käyttöliittymä oli liian monimutkainen käyttäjälle, joka ei tunne taustalla toimivaa ohjelmistoa ja sen toimintaperiaatteita.

Käyttöliittymän tärkeimmät ominaisuudet ovat haku avainsanalla, hakuehtojen muodostaminen (AND, OR ja NOT -operaattoreilla), viestien ajankohdan mukaan rajaaminen ja hakukohteen valitseminen. Ohjelmallisen rajapinnan ja Web-käyttöliittymän tulee olla joustava, jotta koodiltaan uuden analyysin tai hakufiltterin voi liittää helposti järjestelmään.

Seuraavaksi laadittiin luonnosmainen visuaalinen suunnitelma käyttöliittymän ulkoasusta. Luonnokseen aseteltiin kaikki lomakkeen elementit ja mietittiin niiden järjestystä. Ulkoasusta päätettiin tehdä yhdenmukainen muiden moduulien ulkoasujen kanssa.

Helppokäyttöisyyttä parantamaan suunniteltiin ohjeistustekstejä lomakkeen kenttiin, esimerkiksi kun kursorin vie kielikentän ylle, tulee näkyviin lista yleisimmistä kielikoodeista, joita lomakkeessa voidaan käyttää. Toinen vaihtoehto kielen valitsemiseen olisi ollut pudotusvalikko, josta olisi helposti voinut valita joitakin kielikoodeja. Kielivalintaa ei kuitenkaan tehty pudotusvalikoksi, sillä se sulkisi pois harvinaisempien kielien käytön.

Hakua piti pystyä rajaamaan valitsemalla aikaväli, jolla olevat viestit haetaan. Lomakkeeseen lisättiin kentät alkupäivämäärälle ja loppupäivämäärälle. Vanhassa käyttöliittymässä oli ollut se ongelma aikavälin valitsemisen kanssa, että käyttäjä ei saanut mitään ohjeistusta, missä muodossa päivämäärät täytyi syöttää. Uuteen käyttöliittymään lisättiin kalenterinäkymä, joka aukeaa käyttäjän napsauttaessa hiirellä päivämääräkenttiä. Kalenterista voidaan napsauttaa haluttua päivämäärää, jolloin se ilmestyy oikeassa muodossa lomakkeeseen. Kalenterissa on myös rajoitukset sille, kuinka pitkälle ajankohdan pystyy valitsemaan. Jos käyttäjä pystyisi valitsemaan liian laajan aikavälin, haku tulisi liian raskaaksi ja järjestelmä ylikuormittuisi.

Käyttäjän pitää pystyä valitsemaan sosiaalisen median hakurajapinta, josta haku suoritetaan. Vaihtoehtoina on tällä hetkellä Facebook, Twitter ja Topsy. Topsy on eräs sosiaalisen median analysointityökalu, jonka rajapintaa NEMOn ohjelmisto käyttää hyväkseen. Haun tyypin valinta päätettiin tehdä pudotusvalikkona, josta käyttäjä voi valita haluamansa. Vanhassa käyttöliittymässä haun

parametrit olivat erilaiset eri hakurajapinnoilla, mutta päätimme laittaa uuteen versioon jokaiselle samat parametrit selkeyden vuoksi.

Viestejä piti pystyä hakemaan kolmella tavalla: hakusanan, hashtagien ja lähettäjän nimen perusteella. Tästä tehtiin pudotusvalikko hakusanakentän eteen, jossa käyttäjällä on valittavana keyword-, #- ja from-vaihtoehdot. Alkuperäisessä käyttöliittymässä tämä piti osata kirjoittaa samaan kenttään hakusanan kanssa, eikä ohjeistusta ollut.

AND-, OR- ja NOT-operaattoreilla pystytään yhdistelemään hakusanoja ja muodostamaan hakuehtoja. Nämä päätettiin toteuttaa radio-painikkeina joista käyttäjä voi valita yhden. Hakuehtorivejä voi lisätä haluamansa määrän, jolloin voidaan muodostaa monimutkaisempiakin hakuja.

4.2 Toteutus

Louhintaohjelmisto on koodattu Java-ohjelmointikielellä ja sen big data –tietokantana toimii MongoDB. Dynaamisen käyttöliittymän moduuli rakennettiin Drupal 7 –sisällönhallintajärjestelmällä käyttäen JavaScriptiä ja jQuerya. Sivun pohja rakennettiin HTML-kielellä ja ulkoasua säädettiin käyttäen CSS-muotoilua.

Drupal on avoimeen lähdekoodiin perustuva sisällönhallintajärjestelmä, joka soveltuu sekä pienten että suurten ja monimutkaisten verkkosivujen toteuttamiseen. Drupalia voidaan laajentaa moduuleilla, eli eräänlaisilla laajennuspalikoilla, mikä tekee siitä monipuolisen sisällönhallintajärjestelmän. Moduuleita on tarjolla tuhansia erilaisia, ja niitä voi tehdä myös itse.

Projektin toteutus alkoi kehitysympäristön ja Drupalin asentamisella. LAMP-ympäristö ja tarvittavat ohjelmat, kuten Eclipse, asennettiin virtuaalikoneelle. Drupalin asentamisessa ei ollut ongelmia, mutta sen toimintaa piti hieman opetella. Tässä projektissa ei kuitenkaan perehdytä Drupaliin kovin syvällisesti, sillä sisällönhallintajärjestelmä toimi ainoastaan pohjana muulle tekniikalle.

Drupalin perusteisiin tutustumisen jälkeen lähdettiin luomaan moduulia, johon some-louhijan ja big data –tietokannan käyttöliittymää alettiin rakentamaan. Moduulin nimeksi annettiin miner_crawler ja moduulin luomiseksi tehtiin miner_crawler.install, miner_crawler.info ja miner_crawler.module –tiedostot. Näiden tiedostojen tarkoitus on määritellä uusi moduuli Drupalille.

Käyttöliittymän pohjana toimi HTML-sivu, johon luotiin lomakkeen runko taulukon avulla ja määriteltiin lomakkeen kenttien otsikot (kuva 7). Kentät lisättiin dynaamisesti jQuerylla erillisessä JavaScript-tiedostossa. Tämän avulla kenttiä pystytään paremmin hallitsemaan, ja käyttäjä voi lisätä ja poistaa rivejä haluamallaan tavalla.

```
<table class="query-table">
    <thead>
        <tr>
            <th id="query-enabled-head" style="border-radius:10px 0 0 0">Enabled</th>
            <th id="query-type-head">Type</th>
            <th id="query-name-head">Name</th>
            <th id="query-since-head">Since</th>
            <th id="query-until-head">Until</th>
            <th id="query-lang-head">Language</th>
            <th id="query-control-head" style="border-radius:0 10px 0 0" colspan=3 />
        </tr>
    </thead>
    <tbody class="query">
    </tbody>
    <tfoot>
        <tr>
            <th colspan=10 style="border-radius: 0 0 10px 10px; height:10px"></th>
        </tr>
    </tfoot>
</table>
<br>
<button id="save">Save</button>
<button id="reset">Reset</button>
```

Kuva 7. HTML-tiedoston sisältö.

Kun sivun pohja oli luotu, alettiin luoda toiminnallisuutta JavaScriptin ja jQueryn avulla. Lomakkeen kentät lisättiin jQueryn .append()-metodilla. Metodi lisää halutun sisällön viimeisimmäksi elementtiin. Esimerkiksi tässä tapauksessa lisättiin taulukkoon soluja <td>-tagilla (kuva 8).

```
tr.append(
        (jQuery)("<td>").append(
                select3.clone().addClass("query-type")
            )
    );

tr.append(
        (jQuery)("<td>").append(
                (jQuery)("<input>").addClass("query-name").val(params["name"])
            )
    );
tr.append(
        (jQuery)("<td>").append(
                (jQuery)("<input size=7 maxlength=8>").addClass("query-since").val(params["since"])
            )
    );
tr.append(
        (jQuery)("<td>").append(
                (jQuery)("<input size=7 maxlength=8>").addClass("query-until").val(params["until"])
```

Kuva 8. Taulukon solujen lisääminen jQueryn .append()-metodilla.

Koska hakurivejä piti voida lisätä ja poistaa, lisättiin jokaiselle riville + ja – merkit. Näitä napsauttamalla voidaan hallita rivien määrää. Jokaiselle riville laitettiin uniikki tunnistenumero joka toimi laskurityyppisesti, kasvaen automaattisesti aina kun uusia rivejä luodaan. Uniikki tunniste on tärkeä rivejä poistettaessa siksi, että järjestelmä osaa poistaa juuri halutun rivin. Koodissa rivin poistamisfunktio tarkastaa ehtolauseen avulla rivien määrän ennen poistamista, koska käyttäjä ei saa poistaa viimeistä riviä (kuva 9).

```
tr.append(
        (jQuery)("<td>").append(
                (jQuery)
                ("<img src='" + picURL + "sites/all/modules/some_crawler_client/remove.png'/>")
                .attr( "title", "Remove query" ).click(function(e){
                    var row = (jQuery)(this).parent().parent();
                    var qid = row.attr("qid");
                    var all_rows = (jQuery)(".query-row");
                    if ( all_rows.length > 1 ) {
                        var rows = (jQuery)(".query-row[qid='" + qid + "']");
                        for ( var i = 0 ; i < rows.length ; i ++ ) rows[i].remove();
                        var testi = (jQuery)(".query-param[qid='" + qid + "']");
                        for ( var i = 0 ; i < testi.length ; i ++ ) testi[i].remove();
                    }
                }).addClass("op-hover")
            )
    );
```

Kuva 9. Funktio hakurivin poistamiselle.

Tässä käyttöliittymäprojektissa käytettiin kahta jQuery UI:n tarjoamaa funktiota: kalenterityökalua ja ohjetekstejä. JQuery UI on joukko laajennuksia jQueryn JavaScript-kirjastoon, ja sisältää muun muassa erilaisia lisäosia, tehosteita ja teemoja käyttöliittymiä varten. Datepicker ja tooltip –funktiot oli helppo ottaa käyttöön jQuery UI:n asentamisen jälkeen ja ne olivat sopivasti mukautettavissa.

Datepicker-funktiota käytettiin alku- ja päättymispäivämäärän valitsemiseen haulle. Kun käyttäjä napsauttaa hiirellä päivämääräkenttää, aukeaa kalenteri, josta voidaan valita haluttu päivämäärä. Päivämäärä tulee automaattisesti oikeassa muodossa kenttään. Datepicker-funktiota (kuva 10) muokattiin asettamalla muoto, oletuspäivämäärä ja rajoitukset valittavalle aikavälille.

```
217     (jQuery)(".query-since").datepicker({ dateFormat: "dd/mm/y", firstDay: 1 , maxDate: 0 })
218                             .datepicker("setDate", "-7");
219     (jQuery)(".query-until").datepicker({ dateFormat: "dd/mm/y", firstDay: 1 , maxDate: "1m"})
220                             .datepicker("setDate", "0");
```

Kuva 10. Datepicker-funktion määrittelyt.

Tooltip-funktio näyttää ohjetekstin kun kursori viedään halutun elementin ylle. Tämä onnistui lisäämällä funktio sivun koodiin, jonka jälkeen kaikki title-elementtiin kirjoitetut tekstit näkyivät ohjeistuksina. Esimerkiksi yleisimpiä kielikoodeja lisättiin ohjeteksteinä kielivalinnan avuksi (kuva 11).

```
156 ▼ tr.append(
157 ▼       (jQuery)("<td>").append(
158             (jQuery)("<input size=2 maxlength=3 >").addClass("query-language").val(params["language"])
159             .attr( "title", "fi=Finnish, us=English, se=Swedish, ru=Russian" )
160 ⌙       )
161 ⌙   );
```

Kuva 11. Kielivalinnan koodi ja ohjetekstit.

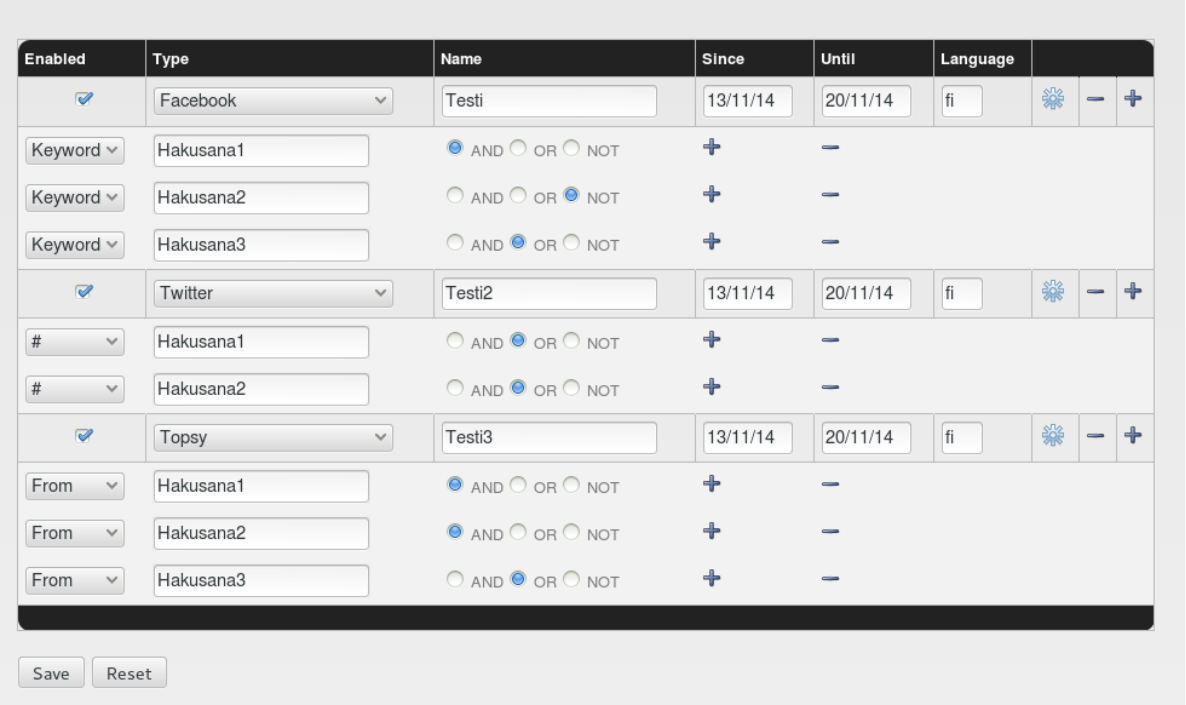
Osaan lomakkeen kentistä asetettiin oletusvaihtoehtoja käyttäjää helpottamaan. Kielen oletusvaihtoehtona on suomi, sillä suurin osa käyttöliittymän käyttäjistä todennäköisesti tekee hakunsa suomeksi. Se antaa myös heti käsityksen siitä, missä muodossa kielen tunnus tulee kirjoittaa, jolloin käyttäjä voi helposti vaihtaa kieltä ymmärrettyään sen logiikan. Haun aikaväliksi asetettiin oletuksena edellinen viikko hakupäivästä laskettuna. Oletusnimi haulle on aina ”unnamed”, jonka käyttäjä voi halutessaan vaihtaa.

Kun käyttäjä napsauttaa save-painiketta, lomakkeeseen täytetyt tiedot kerätään taulukkoon ja lähetetään palvelimelle JQueryn AJAX-kutsun avulla (kuva 12). Tiedot muutetaan JSON-muotoon ennen lähettämistä. JSON eli JavaScript Object Notation on yksinkertainen avoimen standardin tiedostomuoto tiedonvälitystä varten.

```
31      (jQuery)("#save").click(function(e){
32          (jQuery).ajax({
33              type: "POST",
34              url: "http://localhost/SoRa/src/store_query",
35              data: {
36                  query: JSON.stringify(getQuery())
37              }
38          });
39      });
```

Kuva 12. Save-painikkeen koodi, jonka avulla tiedot lähetetään palvelimelle.

Valmis käyttöliittymä (kuva 13) säädettiin näyttämään mahdollisimman selkeältä, sillä usean rivin lomakkeet ovat helposti sekavia. Visuaalista kohinaa saatiin vähennettyä poistamalla turhia viivoja kenttien välissä ja vaihtelemalla taustavärin sävyä eri alueiden erottamiseksi. Käyttäjien kyky sietää monimutkaisuutta vaihtelee, joten ei ole yksiselitteistä tapaa selkeän ulkoasun rakentamiseen. On kuitenkin tärkeää poistaa turhat visuaaliset häiriötekijät.



Kuva 13. Valmis käyttöliittymä.

4.3 Testaus

Koska järjestelmä ja käyttöliittymä toimivat vasta paikallisessa kehitysympäristössä, ei suurempaa käytettävyystestausta voitu vielä järjestää. Käyttöliittymän toimivuutta testattiin kuitenkin NEMO-projektin jäsenten kesken mahdollisimman monessa kehitysvaiheessa. Virheitä korjattiin sitä mukaan kun niitä ilmeni. Lopullinen käytettävyystestaus tullaan järjestämään lähempänä julkaisua.

Funktioiden toimintaan liittyviä ongelmia ratkaistessa käytettiin paljon ponnahdusikkunoita, joiden avulla voitiin testata mihin asti funktio suoriutuu. Ponnahdusikkunan koodi lisättiin haluttuun kohtaan funktion koodia, jolloin se ilmestyy selaimeen jos funktio suoriutuu. Ponnahdusikkunoita käytettiin myös kun testattiin lomakkeen tietojen lähetystä.

Lomakkeen rivien poistamistoiminto tuotti jonkin verran ongelmia, sillä sen testaaminen oli monimutkaisempaa. Funktio saattoi esimerkiksi näyttää toimivalta kun hakuja oli vain yksi, mutta kun hakuja oli enemmän, yhden rivin poistaminen poisti kaikista hauista rivin. Tämä ongelma ratkaistiin yksilöimällä jokainen rivi paremmin.

5 POHDINTA

Projekti onnistui omasta mielestäni tyydyttävästi, ja tavoitteet saavutettiin. Rajallisesta aikataulusta johtuen joitakin elementtejä ei saatu vielä täysin valmiiksi, mutta käyttöliittymä saatiin siihen tilaan että jatkokehitys on mahdollista. NEMO-projekti on suunniteltu suoritettavaksi vuosien 2014-2015 aikana, joten julkaisulla ei ole vielä kiire.

Aikaa käyttöliittymän suunnitteluun ja toteutukseen meni noin kuukausi. Projektiin kulunutta aikaa vei uusien asioiden opettelu, sillä JavaScript, jQuery ja Drupal olivat minulle suhteellisen vieraita elementtejä. Sain kuitenkin tukea NEMO-projektin jäseniltä tarvittaessa, ja he olivat lopputulokseen tyytyväisiä.

Asia jonka olisi voinut vielä tehdä, oli tietokantataulun suunnittelu, jonne tiedot hakulomakkeesta tallennettaisiin. Tällä hetkellä lomake ei varsinaisesti vielä hae mitään, se vasta lähettää siihen kirjoitetut tiedot eteenpäin. Hakutoiminnallisuus lisätään myöhemmin, eikä se ei kuulunut tämän opinnäytetyön toimeksiantoon.

Käyttöliittymä tarvitsee vielä testausta oman kohderyhmänsä parissa ennen kuin voidaan varmistua sen hyvästä käytettävyydestä. On aivan eri asia testata käyttöliittymää alan ammattilaisten kuin vähemmän asiaan perehtyneiden ihmisten kesken. Louhintaohjelmistoa tullaan sen valmistuttua käyttämään tutkimustehtäviin, ja käyttäjäkunta tulee olemaan vaihtelevaa.

LÄHTEET

Ajax Introduction 2014. W3Schools. Viitattu 3.12.2014.
http://www.w3schools.com/ajax/ajax_intro.asp.

Bibeault, B. & Katz, Y. 2010, jQuery in Action. 3., uudistettu painos. Stamford: Manning Publications.

Garret, J. 2005. Ajax: A New Approach to Web Applications. Viitattu 4.12.2014.
http://www.adaptivepath.com/ideas/ajax-new-approach-web-applications/.

How does the Internet work 2014. W3C.Viitattu 28.11.2014.
http://www.w3.org/wiki/How_does_the_Internet_work#Static_vs._Dynamic_Web_Sites.

JQuery Introduction 2014. W3Schools. Viitattu 3.12.2014.
http://www.w3schools.com/jquery/jquery_intro.asp

Karukka, M. & Inkilä, T. 2013. Responsiivinen verkkosivujen suunnittelu mukauttaa sisällön eri päätelaitteille. ePooki. Oulun ammattikorkeakoulun tutkimus- ja kehitystyön julkaisut 6. Viitattu 18.11.2014. http://urn.fi/urn:nbn:fi-fe201302221895.

Korpela, J. 2012. Responsiivinen suunnittelu. Viitattu 18.11.2014.
http://html5kirja.fi/2012/08/02/responsiivinen-suunnittelu/.

Rainio, T. 2011. Dyna Startup, Internet-sivut Ajax- ja jQuery-tekniikkoja käyttäen. Opinnäytetyö. Tietotekniikan koulutusohjelma. Lahti: Lahden ammattikorkeakoulu.

Salovaara, I. 2011. Käyttöliittymäsuunnittelu. Opinnäytetyö. Tietotekniikan koulutusohjelma. Forssa: Hämeen ammattikorkeakoulu.

Sinkkonen, I.; Kuoppala, H.; Parkkinen, J. & Vastamäki, R. 2006. Käytettävyyden psykologia. 3., uudistettu painos. Helsinki: Edita Prima Oy.

Sonninen, K. 2011. Graafisen käyttöliittymän suunnittelu käytettävyyden näkökulmasta. Opinnäytetyö. Tietotekniikan koulutusohjelma. Turku: Turun ammattikorkeakoulu.